IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Dettinger et al.	§	Filed: November 24, 2003
Serial No.: 10/720,960	§	Group Art Unit: 2168
Confirmation No.: 5201	§	Examiner: Mahesh H. Dwivedi

For: DYNAMIC FUNCTIONAL MODULE AVAILABILITY

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Mahesh H. Dwivedi, or electronically transmitted via EFS-Web, on the date shown below:

January 3, 2011 /C. Alexandra Livingston/
Date C. Alexandra Livingston

APPEAL BRIEF

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2168 dated August 2, 2010, finally rejecting claims 10, 14-16, 20, 21, and 27-32. The final rejection of claims 10, 14-16, 20, 21, and 27-32 is appealed. This Appeal Brief is believed to be timely since it is transmitted by the due date of January 1, 2011, as set by the filing of a Notice of Appeal on November 1, 2010.

Please charge the fee of $540.00 for filing this brief to:

Deposit Account No. 09-0465 / ROC920030277US1.

TABLE OF CONTENTS

Real Party in Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicant asserts that no other appeals or interferences are known to the Applicant, the Applicant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 10, 14-16, 20, 21, and 27-32 are pending in the application. Claims 1-29 were originally presented in the application. Claims 30-32 have been added during prosecution. Claims 1-9, 11-13, 17-19, and 22-26 have been cancelled without prejudice. Claims 10, 14-16, 20, 21, and 27-32 stand finally rejected as discussed below. The final rejections of claims 10, 14-16, 20, 21, and 27-32 are appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

All claim amendments have been entered by the Examiner. No amendments to the claims were proposed after the final rejection.

Summary of Claimed Subject Matter

A. CLAIM 10 – INDEPENDENT

Claim 10 is directed to a method of providing a user access to functional modules from within a query application. The method comprises assigning metadata requirements to functional modules that operate on data stored in, or functional modules that generate results that are stored in, a database, wherein the assigned metadata requirements specify conditions required for successful execution of the functional module, wherein at least one condition defines at least one user role required for successful execution of the functional module. Page 12, Li. 8-24; Figure 1, items 129 and 160. The method also comprises collecting run time metadata relating to one or more result fields in a query statement, wherein the result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement, wherein the runtime metadata is collected after composition of the query statement, and wherein the runtime metadata is collected before the query statement is submitted for execution. Page 14, Li. 21-27; Page 15, Li. 1-17; Figure 3A, items 302 and 174. The method further comprises obtaining a list of functional modules that are accessible from within the application used during the query session. Page 15, Li. 24-28; Page 16, Li. 1-12; Figure 3A, items 305 and 129; Figure 3B, item 352. Additionally, the method comprises identifying a limited subset of the functional modules in the list that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements. Page 15, Li. 24-28; Page 16, Li. 1-26; Figure 3A, items 306 and 129_B; Figure 3B, items 353-358. The method also comprises providing an interface presenting the user with the identified limited subset of functional modules that will successfully execute. Page 17, Li. 24-28; Page 18, Li. 1-5; Figure 4C, item 430_C.

B. CLAIM 20 - INDEPENDENT

Claim 20 is directed to a computer readable storage medium containing a program which, when executed, performs operations for providing a user access to functional modules from within an application. The operation comprises assigning metadata requirements to functional modules that operate on data stored in, or functional modules

that generate results that are stored in, a database, wherein the assigned metadata requirements specify conditions required for successful execution of the functional module, wherein at least one condition defines at least one user role required for successful execution of the functional module. Page 12, Li. 8-24; Figure 1, items 129 and 160. The operation also comprises collecting run time metadata relating to one or more result fields in a query statement, wherein the result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement, wherein the runtime metadata is collected after composition of the query statement, and wherein the runtime metadata is collected before the query statement is submitted for execution. Page 14, Li. 21-27; Page 15, Li. 1-17; Figure 3A, items 302 and 174. The operation further comprises obtaining a list of functional modules that are accessible from within the application used during the query session. Page 15, Li. 24-28; Page 16, Li. 1-12; Figure 3A, items 305 and 129; Figure 3B, item 352. Additionally, the operation comprises identifying a limited subset of the functional modules in the list that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements. Page 15, Li. 24-28; Page 16, Li. 1-26; Figure 3A, items 306 and 129_B; Figure 3B, items 353-358. The operation also comprises providing an interface presenting the user with the identified limited subset of functional modules that will successfully execute. Page 17, Li. 24-28; Page 18, Li. 1-5; Figure 4C, item 430_C.

C. CLAIM 27 INDEPENDENT

Claim 27 is directed to a data processing system for providing a user access to functional modules from within an application. The data processing system generally comprises a data repository and a plurality of functional modules, each having associated metadata requirements that specify conditions required for successful execution of the functional modules, wherein at least one condition defines at least one user role required for successful execution of the functional modules. Page 12, Li. 8-24; Figure 1, items 156, 129, and 160. The data processing system further comprises an application from which the functional modules are accessible, wherein the application is configured to, after composition of a query statement, but before the query statement is

submitted for execution, collect runtime metadata related to one or more result fields in the query statement, wherein the one or more result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement. Page 14, Li. 21-27; Page 15, Li. 1-17; Figure 3A, items 302 and 174. The application is further configured to present to a user a limited subset of the functional modules that will successfully execute, as determined by the application based on the collected runtime metadata and the metadata requirements associated with the functional modules. Page 15, Li. 24-28; Page 16, Li. 1-26; Page 17, Li. 24-28; Page 18, Li. 1-5; Figure 3A, items 306 and 129_B; Figure 3B, items 353-358; Figure 4C, item 430_C.

Grounds of Rejection to be Reviewed on Appeal

1. Rejection of claims 10, 14-16, 20-21, and 27-32 under 35 U.S.C. § 103(a) as being unpatentable over *Win et al.,* (US 6,453,353, hereinafter *Win*), in view of *Faybishenko et al*., (US 2003/0158839, hereinafter *Faybishenko*), and further in view of *Chang et al.*, (US 6,968,509), hereinafter *Chang*.

ARGUMENTS

1. Rejection of claims 10, 14-16, 20-21, and 27-32 under 35 U.S.C. § 103(a) as being unpatentable over *Win,* in view of *Faybishenko*, and further in view of *Chang*.

The Applicable Law

The Examiner bears the initial burden of establishing a prima facie case of obviousness. See MPEP § 2141. Establishing a prima facie case of obviousness begins with first resolving the factual inquiries of Graham v. John Deere Co. 383 U.S. 1 (1966). The factual inquiries are as follows:

(A) determining the scope and content of the prior art;

(B) ascertaining the differences between the claimed invention and the prior art;

(C) resolving the level of ordinary skill in the art; and

(D) considering any objective indicia of nonobviousness.

Once the Graham factual inquiries are resolved, the Examiner must determine whether the claimed invention would have been obvious to one of ordinary skill in the art.

The Examiner's Argument

The Examiner conceeds that "*Win* does not explicitly teach … collecting runtime metadata relating to one or more result fields in a query statement … wherein the one or more result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement … [and] wherein the runtime metadata is collected after composition of the query statement." However, the Examiner states that *Faybishenko* discloses these claimed limitations, and that "it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of the cited references because teaching *Faybishenko's* would have allowed *Win's* to provide a method to allow for more control over content provided over

the internet to providers." Additionally, while the Examiner admits that neither *Win* nor *Faybishenko* teaches collecting runtime metadata before the query statement is submitted for execution, the Examiner suggests that *Chang* teaches such a limitation, and argues that "it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the teachings of the cited references because teaching *Chang's* would have allowed *Win's* and *Faybishenko's* to provide a method that adequately records user events." Respectfully, Applicants disagree.

Applicants' Response to the Examiner's Argument

Applicants submit that *Dogl* does not disclose "each and every element as set forth in the claim". For example, *Dogl* does not disclose a method of providing a user access to functional modules from within a query application that includes "collecting runtime metadata relating to one or more result fields in a query statement, wherein the one or more result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement." The Examiner suggests that the following passage from *Faybishenko* teaches this limitation: "In one embodiment a QRP adapter may monitor or log queries, results, number of hits, searches, results, etc. or generally the information passing through the QRP adapter." *Faybishenko,* [0112]. Thus, *Faybishenko* teaches monitoring or logging search traffic (including search queries, search results, the number of hits for a search query, etc.) passing over a network and through a particular network adapter (i.e., the QRP adapter). *Faybishenko* continues, stating:

> [A] user interface may be provided through which providers may view the results of searches and hits performed by consumers – e.g. how many searches resulted in their entry being returned, how many users clicked through, etc. In one embodiment, a user interface may be provided through which providers may monitor and/or control the number of queries sent to them and also to throttle traffic (e.g., turn it off) if necessary.

Faybishenko at Para. [0112]. In other words, the cited passage teaches monitoring search traffic generally and, based on the monitored information, displaying the results and number of the searches.

However, *Faybishenko* fails to teach the claimed limitation of "collecting runtime metadata relating to one or more result fields in a query statement, wherein the one or more result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement." That is, while the cited passages from *Faybishenko* teach that a submitted query passing across a network adapter (i.e., the QRP adapter) may be monitored and/or logged, such a teaching pertains the query statement itself, rather than "metadata relating to one or more result fields in a query statement." Thus, even if, as the Examiner contends, Faybishenko teaches "logging an entire query … [where] the result fields are encompassed within that query," such a teaching still fails to teach the claimed limitation of "collecting runtime metadata relating to one or more result fields in a query statement."

Furthermore, the current claim also recites "collecting runtime metadata … wherein the runtime metadata is collected after composition of the query statement, and wherein the runtime metadata is collected before the query statement is submitted for execution." Thus, in other words, the "collecting…" step of the current claim occurs at a moment in time after the query has been composed, but before the query has been submitted for execution. The Examiner cites to two references as teaching such a period of time: *Faybishenko* for teaching the limitation of "collecting runtime metadata … after composition of the query statement," and *Chang* for teaching the limitation of "collecting runtime metadata … before the query statement is submitted for execution."

As discussed above, *Faybishenko* teaches a network adapter (i.e., the QRP adapter) that monitors traffic passing through it, and that such traffic may include a submitted query. Thus, while the logging of *Faybishenko* may occur after query composition, such logging also occurs after the query has been submitted for execution. That is, because *Faybishenko* teaches monitoring a query that is passing through a network adapter, such monitoring occurs after the query has been submitted for execution. As such, the monitoring and logging taught by *Faybishenko* occur after both composition and submission of the query.

Nonetheless, the Examiner argues that *Chang* teaches the claimed limitation of "collecting runtime metadata … before the query statement is submitted for execution." More specifically, the Examiner states: "*Chang* records, saves, and displays all

concurrent typed keyboard strokes by a user. Because *Chang* records the typed keyboard strokes as a user is typing them, then, as a result, the typed keywords are saved before a 'submission of execution.'" As an initial note, Applicants note that *Chang* does not teach "collecting runtime metadata relating to one or more result fields in a query statement" at any point in time, but merely teaches recording and displaying keyboard strokes as a user types them. Nonetheless, any "collecting" taught by *Chang* would occur not only before submission of a query, but before composition of the query as well. At best, *Chang* would merely teach recording the text a user types during query composition.

Based on these teachings, the Examiner suggests that the combination of *Faybishenko* and *Chang* teaches the claimed limitation of "collecting runtime metadata relating to one or more result fields in a query statement … after composition of the query statement, and … before the query statement is submitted for execution." However, as demonstrated abov, the combination of *Faybishenko* and *Chang* teaches logging the text of a query statement before both composition and submission of the query statement, or after both composition and submission of the query statement. Thus, the combination of the cited references would at best teach collecting metadata before or after the claimed period of time (i.e., after composition of the query statement but before the query statement is submitted for execution), but fails to teach collecting metadata during the claimed period of time itself. Accordingly, Applicants submit that the combination of *Win* in view of *Faybishenko* and further in view of *Chang* fails to disclose all of the elements of the current claim 10. As such, claim 10 is believed to be allowable, and allowance is respectfully requested.

Claims 20 and 27 contain substantially the same limitations as claim 10, and the Examiner rejects these claims for substantially the same reasons. Since claim 10 is believed to be allowable, claims 20, 27 and all dependent claims are allowable as well. Therefore, all claims are believed to be allowable, and allowance of the claims is respectfully requested.

CONCLUSION

The Examiner errs in finding that claims 10, 14-16, 20-21, and 27-32 are unpatentable over *Win*, in view of *Faybishenko*, and further in view of *Chang* under 35 U.S.C. § 103(a).

Withdrawal of the rejections and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. MCCLELLAN, Reg. #44,227/

Gero G. McClellan
Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
1150 Revolution Mill Drive, Studio 5
Greensboro, NC 27405
Telephone: (336) 698-4286
Facsimile: (713) 623-4846
Attorney for Appellant(s)

CLAIMS APPENDIX

1-9. (Cancelled)

10. (Previously Presented) A method of providing a user access to functional modules from within a query application, comprising:

assigning metadata requirements to functional modules that operate on data stored in, or functional modules that generate results that are stored in, a database, wherein the assigned metadata requirements specify conditions required for successful execution of the functional module, wherein at least one condition defines at least one user role required for successful execution of the functional module;

collecting runtime metadata relating to one or more result fields in a query statement, wherein the one or more result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement, wherein the runtime metadata is collected after composition of the query statement, and wherein the runtime metadata is collected before the query statement is submitted for execution;

obtaining a list of functional modules that are accessible from within an application used during the query session;

identifying a limited subset of the functional modules in the list that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements; and

providing an interface presenting the user with the identified limited subset of functional modules that will successfully execute.

11-13. (Cancelled)

14. (Previously Presented) The method of claim 10, wherein obtaining metadata associated with the functional module comprises examining a signature validation.

15. (Previously Presented) The method of claim 10, wherein the metadata associated with at least one of the functional modules comprises at least one of: one or

more input parameters required for successful execution of the functional module, one or more output parameters required for successful execution of the functional module, and a security credential required to execute the functional module.

16. (Original) The method of claim 10, wherein at least one of the functional modules analyzes query results.

17-19. (Cancelled)

20. (Previously Presented) A computer readable storage medium containing a program which, when executed, performs operations for providing a user access to functional modules from within an application, comprising:

assigning metadata requirements to functional modules that operate on data stored in, or functional modules that generate results that are stored in, a database, wherein the assigned metadata requirements specify conditions required for successful execution of the functional module, wherein at least one condition defines at least one user role required for successful execution of the functional module;

collecting runtime metadata relating to one or more result fields in a query statement, wherein the one or more result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement, wherein the runtime metadata is collected after composition of the query statement, and wherein the runtime metadata is collected before the query statement is submitted for execution;

obtaining a list of functional modules accessible from within the application;

identifying a limited subset of functional modules that will successfully execute, by comparing the collected runtime metadata with the assigned metadata requirements; and

providing an interface presenting the user with the identified limited subset of functional modules that will successfully execute.

21. (Previously Presented) The computer readable storage medium of claim 20 wherein the application is a query building application.

22-26. (Cancelled)

27. (Previously Presented) A data processing system for providing a user access to functional modules from within an application comprising:

a data repository;

a plurality of functional modules, each having associated metadata requirements that specify conditions required for successful execution of the functional modules, wherein at least one condition defines at least one user role required for successful execution of the functional modules;

an application from which the functional modules are accessible, wherein the application is configured to:

after composition of a query statement, but before the query statement is submitted for execution, collect runtime metadata related to one or more result fields in the query statement, wherein the one or more result fields specify one or more data fields for which data is requested to be returned upon execution of the query statement; and

present to a user a limited subset of the functional modules that will successfully execute, as determined by the application based on the collected runtime metadata and the metadata requirements associated with the functional modules.

28. (Original) The data processing system of claim 27, wherein the data repository comprises XML data structures used to store runtime metadata.

29. (Original) The data processing system of claim 27, wherein the data repository comprises relational database tables used to store runtime metadata.

30. (Previously Presented) The method of claim 10, wherein the runtime metadata relating to one or more result fields in the query statement comprises one or more of:

a result field name; and

a data type for the result field.

31. (Previously Presented) The computer readable storage medium of claim 20, wherein the runtime metadata relating to one or more result fields in the query statement comprises one or more of:

a result field name; and

a data type for the result field.

32. (Previously Presented) The data processing system of claim 27, wherein the runtime metadata related to one or more result fields in the query statement comprises one or more of:

a result field name; and

a data type for the result field.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.